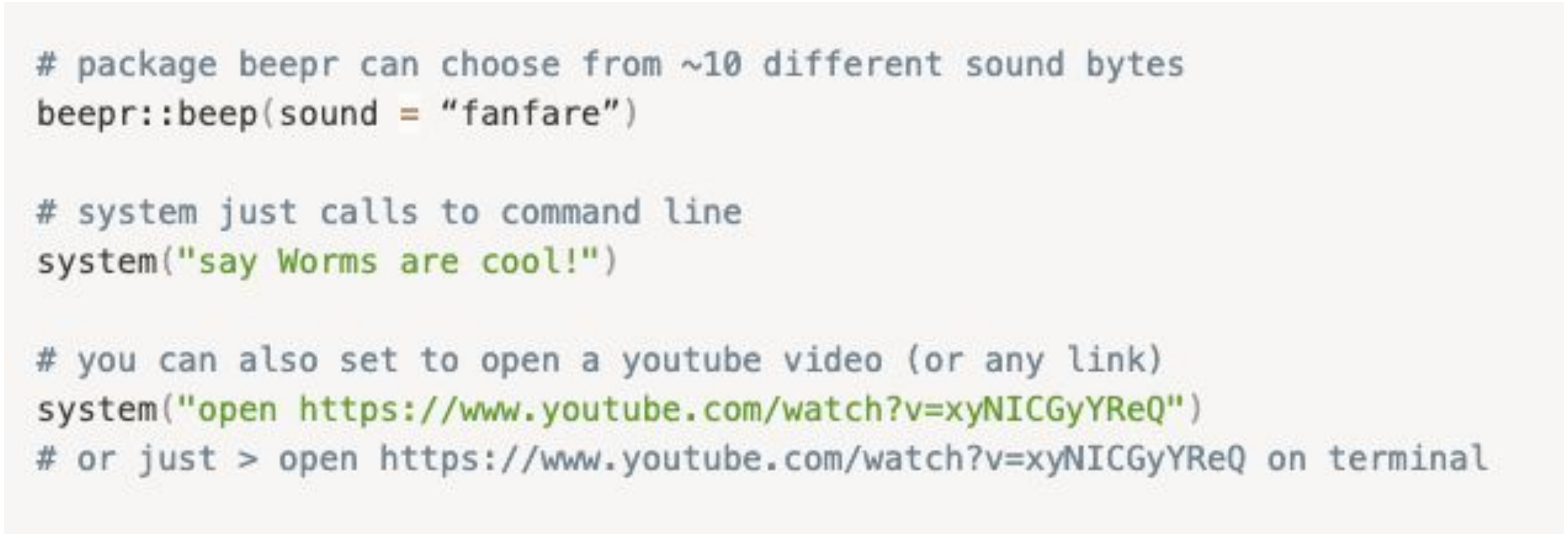
Any general code comments, questions, tips/tricks?

R can play sound, text-to-talk, or open a youtube video?



```r
# package beepr can choose from ~10 different sound bytes
beepr::beep(sound = “fanfare”)

# system just calls to command line
system("say Worms are cool!")

# you can also set to open a youtube video (or any link)
system("open https://www.youtube.com/watch?v=xyNICGyYReQ")
# or just > open https://www.youtube.com/watch?v=xyNICGyYReQ on terminal
```

Or just download youtube videos with terminal: here

Git/Github authentication setup?

Set credentials on Mac:

```
git config --global credential.helper osxkeychain
```

Erase credentials on Mac:

```
git credential-osxkeychain erase
host=github.com
protocol=https
```

Need a personal authentication token: here

Git/Github authentication setup?

Set credentials on QUEST:

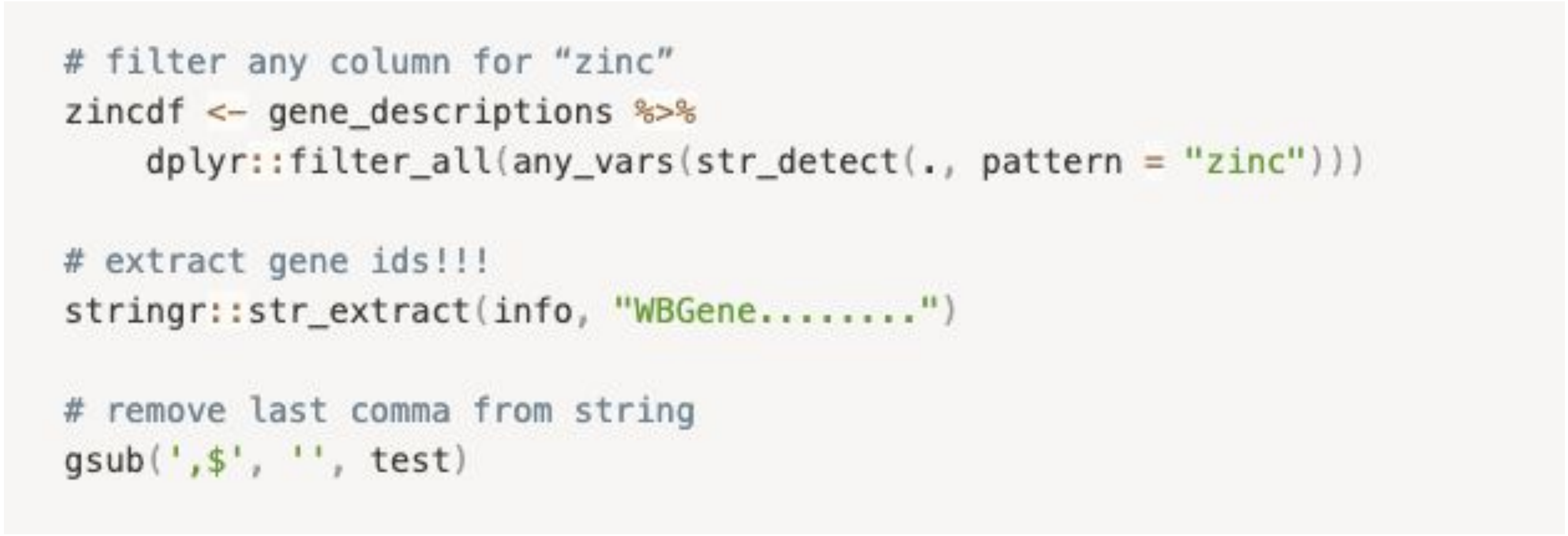
```
git config --global credential.helper cache
```

Erase credentials on QUEST:

```
git config --global --unset credential.helper
```

Need a personal authentication token: here

Other misc. finds



```
# filter any column for “zinc”
zincdf <- gene_descriptions %>%
    dplyr::filter_all(any_vars(str_detect(., pattern = "zinc")))

# extract gene ids!!!
stringr::str_extract(info, "WBGene........")

# remove last comma from string
gsub(',$', '', test)
```

Maybe next time we should talk about ***`stringr`****?*

Running nextflow pipelines with GitHub

Local

1. Clone git repo to QUEST

```
git clone https://github.com/AndersenLab/NemaScan.git
cd NemaScan
```

2. Run pipeline

```
nextflow run develop.nf --debug
```

Remote

1. Run pipeline remotely using git repo name

```
nextflow run AndersenLab/NemaScan/develop.nf --debug
```

Running nextflow pipelines with GitHub

Local

PRO

- Only update pipeline when you manually `git pull` (control)
- Easy to make changes to suit your specific needs

CON

- Might end up with several different versions of the same pipeline
- Might not know if there is a new update

Remote

PRO

- Always get newest version (but can still run older versions)
- Nextflow logs git commit - reproducible versions
- Can run anywhere on QUEST

CON

- Will still need to clone/branch to make specific edits

How to run a specific git branch/commit with nextflow

# run master branch, newest commit

```
nextflow run AndersenLab/NemaScan --debug
```

# run specific commit (maybe a previous version)

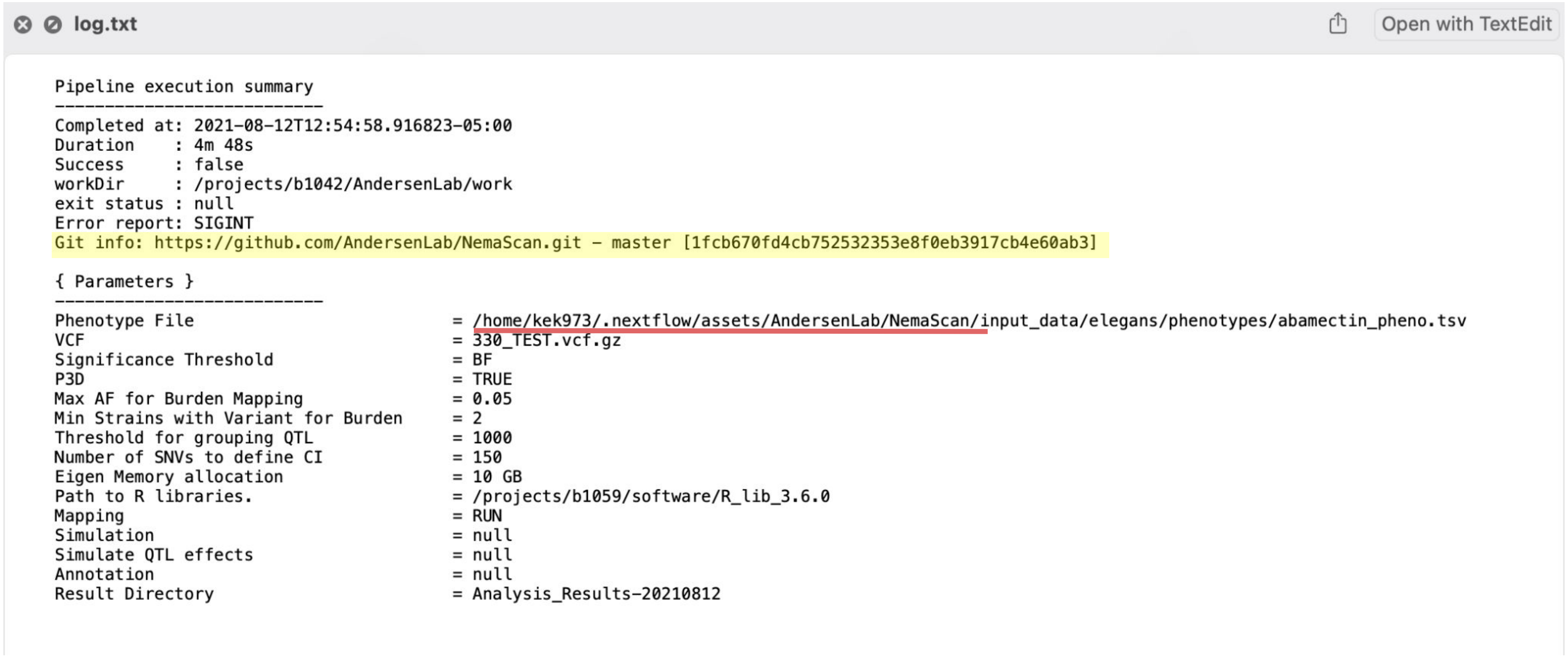
```
nextflow run andersenlab/nemascan --debug -r <commitid>
```

# run newest commit on specific branch

```
nextflow run AndersenLab/NemaScan --debug -r cendr_dev
```

➔ *More info: here*

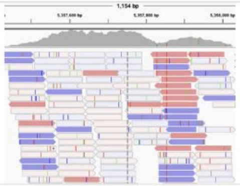
Why would I want to run nextflow remote?



log.txt Open with TextEdit

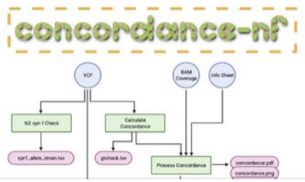
```
Pipeline execution summary
---------------------------
Completed at: 2021-08-12T12:54:58.916823-05:00
Duration    : 4m 48s
Success     : false
workDir     : /projects/b1042/AndersenLab/work
exit status : null
Error report: SIGINT
Git info: https://github.com/AndersenLab/NemaScan.git - master [1fcb670fd4cb752532353e8f0eb3917cb4e60ab3]

{ Parameters }
---------------------------
Phenotype File                          = /home/kek973/.nextflow/assets/AndersenLab/NemaScan/input_data/elegans/phenotypes/abamectin_pheno.tsv
VCF                                     = 330_TEST.vcf.gz
Significance Threshold                  = BF
P3D                                     = TRUE
Max AF for Burden Mapping               = 0.05
Min Strains with Variant for Burden     = 2
Threshold for grouping QTL              = 1000
Number of SNVs to define CI             = 150
Eigen Memory allocation                 = 10 GB
Path to R libraries.                    = /projects/b1059/software/R_lib_3.6.0
Mapping                                 = RUN
Simulation                              = null
Simulate QTL effects                    = null
Annotation                              = null
Result Directory                        = Analysis_Results-20210812
```

What pipelines *should* I run remote?

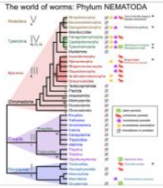


Alignment-nf

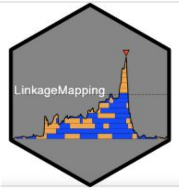
cegwas2-nf



concordance-nf



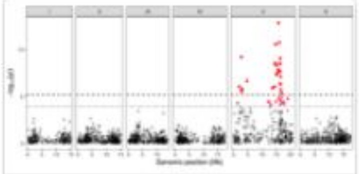
genomes-nf



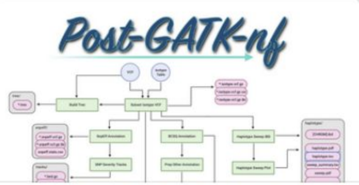
linkagemapping-nf (a...

```
parameters          description                    Set/Default
==========          ===========                    =======

--debug             Set to 'true' to test          false
--cores             Number of cores                4
--A                 Parent A                       N2
--B                 Parent B                       CB4856
--cA                Parent A color (for plots)     #0000FF
--cB                Parent B color (for plots)     #FF0000
--out               Directory to output results    NIL-N2-CB4856-2017-09-27
--fqs               fastq file (see help)          (required)
--relative          use relative fastq prefix      ${params.relative}
--reference         Reference Genome               /Users/dancook/Documents/git/nil-nf/refe
--vcf               VCF to fetch parents from      (required)
--tmpdir            A temporary directory          tmp/
```

nil-ril-nf



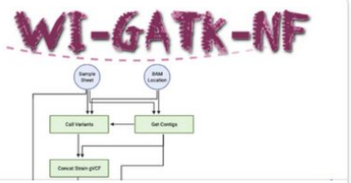
Nemascan



post-gatk-nf

```
parameters              description                                  Set/Default
==========              ===========                                  ========================
--debug                 Use --debug to indicate debug mode           ${params.debug}
--fastq_folder          Name of the raw fastq folder                 ${params.fastq_folder}
--raw_path              Path to raw fastq folder                     ${params.raw_path}
--processed_path        Path to processed fastq folder (output)      ${params.processed_path}
--trim_only             Whether to skip species check and only trim  ${params.trim_only}
--genome_sheet          File with fasta locations for species check  ${params.genome_sheet}
--out                   Folder name to write results                 ${params.out}
--subsample_read_count  How many reads to use for species check      ${params.subsample_read_co
```

trim-fq-nf



wi-gatk

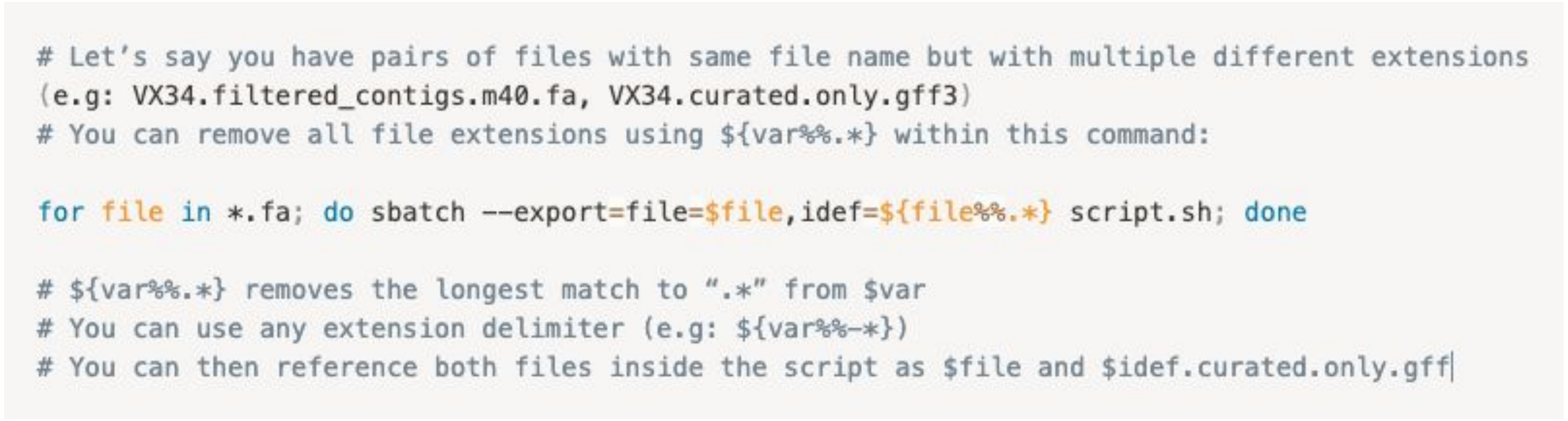
Passing filename without file extension as SBATCH variable

```
# You have a set of files with the same file extension
# (e.g.: VX34.fa)
# You want to run a script on each one of this files

for file in *.fa; do sbatch --export=file=$file script.sh; done
```

```
# Let's say that you have pairs of files with same file name but different extensions
# (e.g.: VX34.fa, VX34.gff)
# You can remove the last file extension using ${var%.*} within this command:

for file in *.fa; do sbatch --export=file=${file%.*} script.sh; done

# ${var%.*} removes the shortest match to ".*" from $var
# You can use any extension delimiter (e.g: ${var%-*})
# You can then reference both files inside the script as $file.fa and $file.gff
```

Credit: Nic

Passing filename without file extension as SBATCH variable



```
# Let's say you have pairs of files with same file name but with multiple different extensions
(e.g: VX34.filtered_contigs.m40.fa, VX34.curated.only.gff3)
# You can remove all file extensions using ${var%%.*} within this command:

for file in *.fa; do sbatch --export=file=$file,idef=${file%%.*} script.sh; done

# ${var%%.*} removes the longest match to ".*" from $var
# You can use any extension delimiter (e.g: ${var%%-*})
# You can then reference both files inside the script as $file and $idef.curated.only.gff
```

Credit: Nic

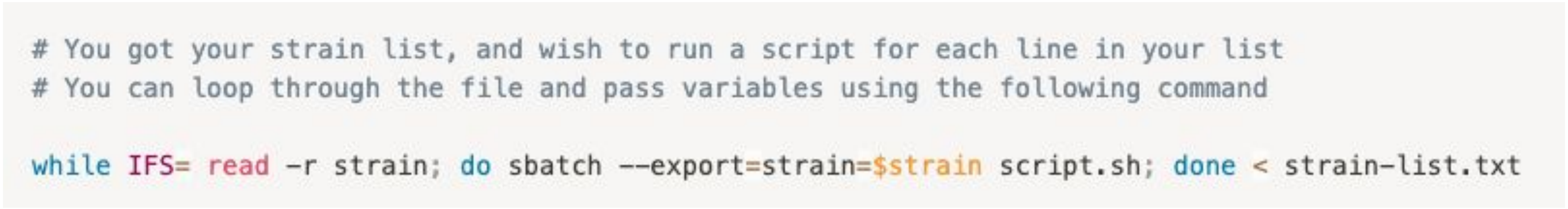
Using the previous expression to generate lists

```
# Let’s say you have a folder full of FASTQ, and you want to generate a strain list from that folder.
# Most FASTQ files in our storage are formatted as ‘STRAIN_[...].fq.gz’
# -(e.g: XZ1733_CKDL200150306-1a-GA03-AK1863_HV53CDSXX_L3_2P.fq.gz)
# You can write a strain list using the ${var%%_*} within this command:

for file in *.fq.gz; do echo ${file%%_*}; done > strain-list.txt
```

```
# If you are expecting duplicates (different lanes for the same strain) you can pipe ‘sort -u’:

for file in *.fq.gz; do echo ${file%%_*}; done | sort -u > strain-list.txt
```

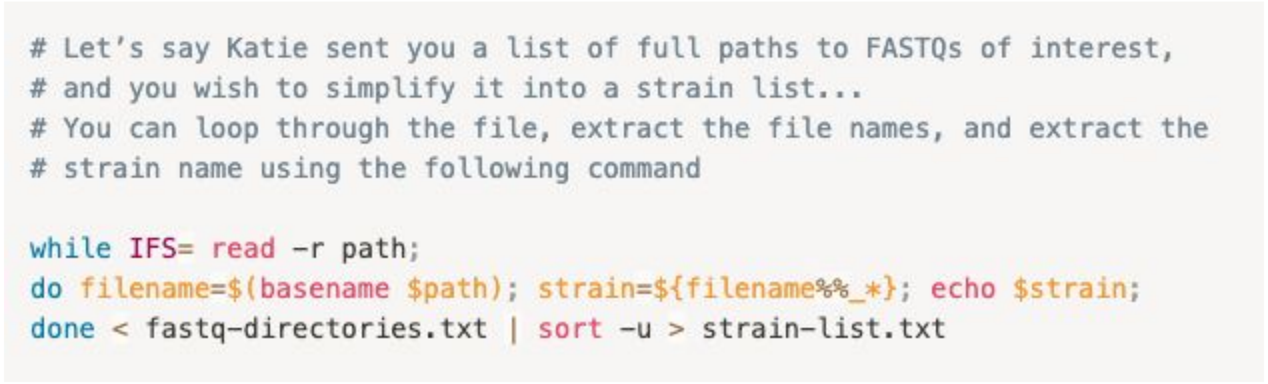
Credit: Nic

Passing variable from list as SBATCH variable



```
# You got your strain list, and wish to run a script for each line in your list
# You can loop through the file and pass variables using the following command

while IFS= read -r strain; do sbatch --export=strain=$strain script.sh; done < strain-list.txt
```

Generate strain list from from directory list



```
# Let's say Katie sent you a list of full paths to FASTQs of interest,
# and you wish to simplify it into a strain list...
# You can loop through the file, extract the file names, and extract the
# strain name using the following command

while IFS= read -r path;
do filename=$(basename $path); strain=${filename%%_*}; echo $strain;
done < fastq-directories.txt | sort -u > strain-list.txt
```

Credit: Nic

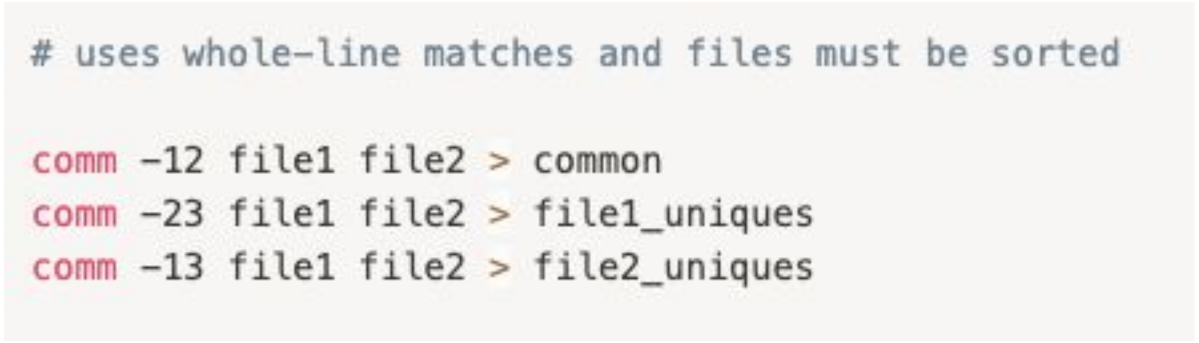
Subsetting with grep (different file types)

```
# Let’s say you want to subset a file based on whole-word matches from a list
# -e.g.: subsetting your sample sheet based on a list of strainst

grep -wFf subset.txt all.txt > all_subset.txt
```

```
# or if applied to the example above:

grep -wFf strains.txt sample_sheet.tsv > subset_ss.tsv

# This will return every line in ‘sample_sheet.tsv’ that has a whole-word match
# (literal substring) to each line in ‘strains.txt’
# Doesn’t require sorted files, and lines can have different fields
```

Credit: Nic

Subsetting & pattern matching with comm (same file types)

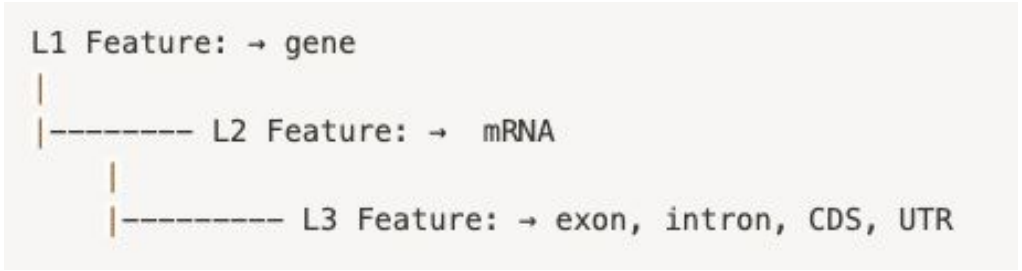


```
# uses whole-line matches and files must be sorted

comm -12 file1 file2 > common
comm -23 file1 file2 > file1_uniques
comm -13 file1 file2 > file2_uniques
```

Credit: Nic

Manipulating wormbase GFFs with grep & sed

- GFF annotations are flat files that have the following structure for protein coding genes:



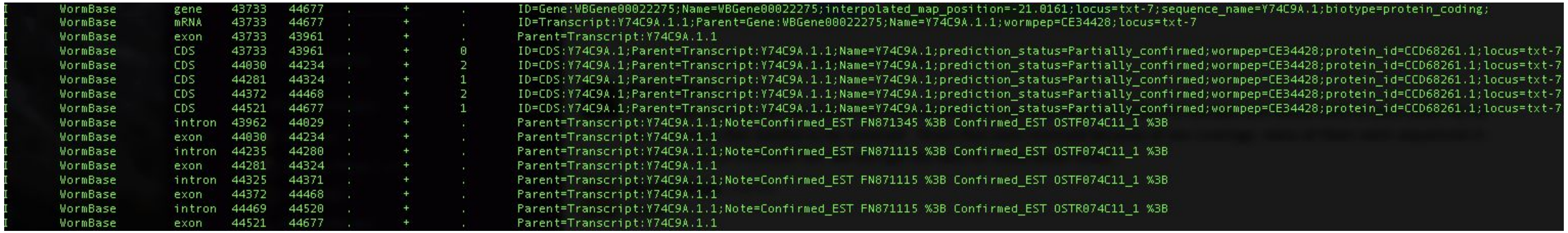
```
L1 Feature: → gene
|
|-------- L2 Feature: →  mRNA
    |
    |--------- L3 Feature: → exon, intron, CDS, UTR
```

- Low level features are nested within higher level features, with genes being at the top
- Genes have different ‘Attributes’ than lower level features. The Attributes field points to the parent in lower level features, and provides additional information about that gene in L1 features.

Credit: Nic

Manipulating wormbase GFFs with grep & sed

- Sample gene in GFF:



```
I	WormBase	gene	43733	44677	.	+	.	ID=Gene:WBGene00022275;Name=WBGene00022275;interpolated_map_position=-21.0161;locus=txt-7;sequence_name=Y74C9A.1;biotype=protein_coding;
I	WormBase	mRNA	43733	44677	.	+	.	ID=Transcript:Y74C9A.1.1;Parent=Gene:WBGene00022275;Name=Y74C9A.1.1;wormpep=CE34428;locus=txt-7
I	WormBase	exon	43733	43961	.	+	.	Parent=Transcript:Y74C9A.1.1
I	WormBase	CDS	43733	43961	.	+	0	ID=CDS:Y74C9A.1;Parent=Transcript:Y74C9A.1.1;Name=Y74C9A.1;prediction_status=Partially_confirmed;wormpep=CE34428;protein_id=CCD68261.1;locus=txt-7
I	WormBase	CDS	44030	44234	.	+	2	ID=CDS:Y74C9A.1;Parent=Transcript:Y74C9A.1.1;Name=Y74C9A.1;prediction_status=Partially_confirmed;wormpep=CE34428;protein_id=CCD68261.1;locus=txt-7
I	WormBase	CDS	44281	44324	.	+	1	ID=CDS:Y74C9A.1;Parent=Transcript:Y74C9A.1.1;Name=Y74C9A.1;prediction_status=Partially_confirmed;wormpep=CE34428;protein_id=CCD68261.1;locus=txt-7
I	WormBase	CDS	44372	44468	.	+	2	ID=CDS:Y74C9A.1;Parent=Transcript:Y74C9A.1.1;Name=Y74C9A.1;prediction_status=Partially_confirmed;wormpep=CE34428;protein_id=CCD68261.1;locus=txt-7
I	WormBase	CDS	44521	44677	.	+	1	ID=CDS:Y74C9A.1;Parent=Transcript:Y74C9A.1.1;Name=Y74C9A.1;prediction_status=Partially_confirmed;wormpep=CE34428;protein_id=CCD68261.1;locus=txt-7
I	WormBase	intron	43962	44029	.	+	.	Parent=Transcript:Y74C9A.1.1;Note=Confirmed_EST FN871345 %3B Confirmed_EST OSTF074C11_1 %3B
I	WormBase	exon	44030	44234	.	+	.	Parent=Transcript:Y74C9A.1.1
I	WormBase	intron	44235	44280	.	+	.	Parent=Transcript:Y74C9A.1.1;Note=Confirmed_EST FN871115 %3B Confirmed_EST OSTF074C11_1 %3B
I	WormBase	exon	44281	44324	.	+	.	Parent=Transcript:Y74C9A.1.1
I	WormBase	intron	44325	44371	.	+	.	Parent=Transcript:Y74C9A.1.1;Note=Confirmed_EST FN871115 %3B Confirmed_EST OSTF074C11_1 %3B
I	WormBase	exon	44372	44468	.	+	.	Parent=Transcript:Y74C9A.1.1
I	WormBase	intron	44469	44520	.	+	.	Parent=Transcript:Y74C9A.1.1;Note=Confirmed_EST FN871115 %3B Confirmed_EST OSTR074C11_1 %3B
I	WormBase	exon	44521	44677	.	+	.	Parent=Transcript:Y74C9A.1.1
```

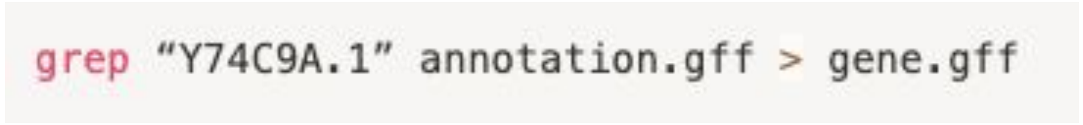
- If you tried to use grep to pull the GFF features using a wormbase ID for a given gene, it will only return the L1/L2 features, but none of the L3 features

```
grep “WBGene00022275” annotation.gff > gene.gff
```

Credit: Nic

Manipulating wormbase GFFs with grep & sed

- A simple solution would be to find a common attribute across all features (like gene ID), and pull all lines pertaining to that gene

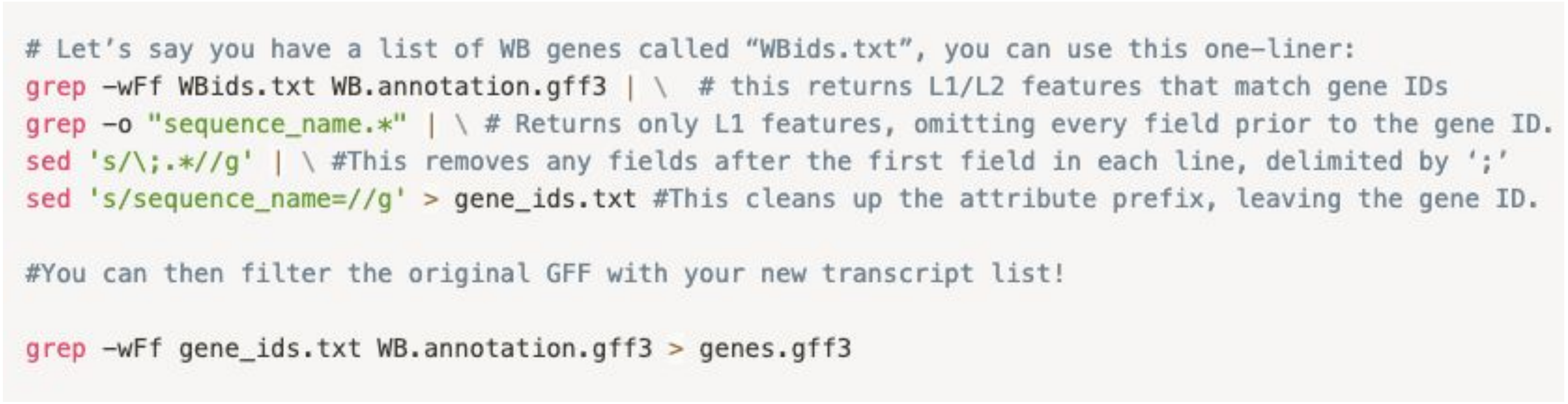


```
grep “Y74C9A.1” annotation.gff > gene.gff
```

- This would work great for a single gene, but what if you had a massive list of WB genes? Too much manual work…

Credit: Nic

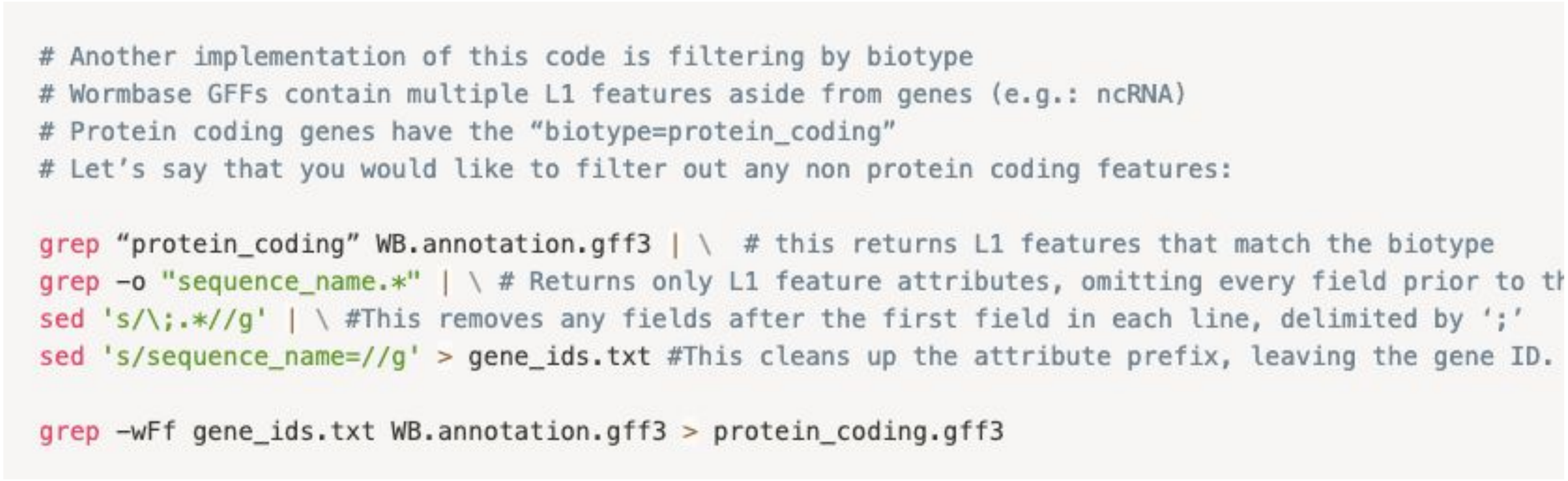
Manipulating wormbase GFFs with grep & sed



```
# Let’s say you have a list of WB genes called “WBids.txt”, you can use this one-liner:
grep -wFf WBids.txt WB.annotation.gff3 | \  # this returns L1/L2 features that match gene IDs
grep -o "sequence_name.*" | \ # Returns only L1 features, omitting every field prior to the gene ID.
sed 's/\;.*//g' | \ #This removes any fields after the first field in each line, delimited by ‘;’
sed 's/sequence_name=//g' > gene_ids.txt #This cleans up the attribute prefix, leaving the gene ID.

#You can then filter the original GFF with your new transcript list!

grep -wFf gene_ids.txt WB.annotation.gff3 > genes.gff3
```

Credit: Nic

Manipulating wormbase GFFs with grep & sed



```
# Another implementation of this code is filtering by biotype
# Wormbase GFFs contain multiple L1 features aside from genes (e.g.: ncRNA)
# Protein coding genes have the “biotype=protein_coding”
# Let’s say that you would like to filter out any non protein coding features:

grep “protein_coding” WB.annotation.gff3 | \  # this returns L1 features that match the biotype
grep -o "sequence_name.*" | \ # Returns only L1 feature attributes, omitting every field prior to th
sed 's/\;.*//g' | \ #This removes any fields after the first field in each line, delimited by ‘;’
sed 's/sequence_name=//g' > gene_ids.txt #This cleans up the attribute prefix, leaving the gene ID.

grep -wFf gene_ids.txt WB.annotation.gff3 > protein_coding.gff3
```

Credit: Nic